บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 แนวคิดและทฤษฎี

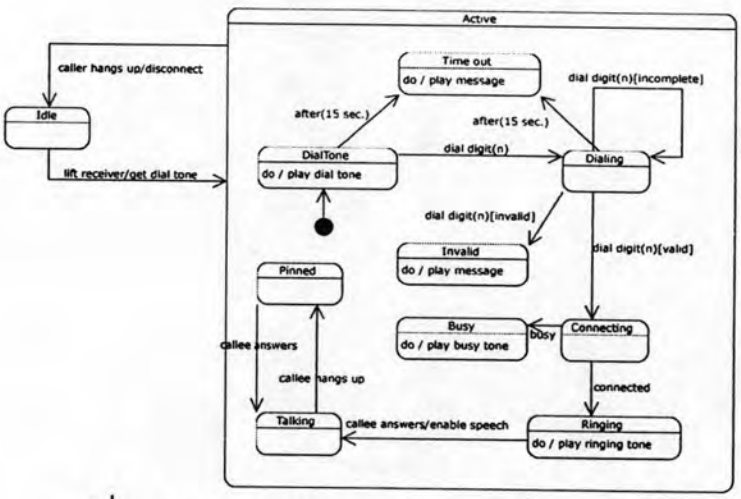
2.1.1 แผนภาพสถานะ (Statechart Diagram)

แผนภาพยูเอ็มแอลประกอบด้วยแผนภาพทั้งหมด 9 แผนภาพ คือ

- แผนภาพคลาส (class diagram) ใช้สำหรับแสดงข้อมูลคลาส ส่วนต่อประสาน (interface) และการเชื่อมต่อกัน ซึ่งเป็นแผนภาพสำคัญสำหรับการพัฒนาซอฟต์แวร์ในระบบเชิงวัตถุ สามารถนำไปเป็นต้นแบบโครงสร้างในการเขียนรหัสโปรแกรมได้
- แผนภาพวัตถุ (object diagram) ใช้สำหรับแสดงข้อมูลวัตถุ และความสัมพันธ์ระหว่างกัน
- แผนภาพยูสเคส (use case diagram) ใช้สำหรับแสดงข้อมูลยูสเคส (use cases) ผู้แสดง (actors) และความสัมพันธ์ระหว่างกัน ซึ่งช่วยอธิบายความต้องการของผู้ใช้ที่มีต่อระบบในระดับสูง
- แผนภาพลำดับ (sequence diagram) ใช้สำหรับแสดงการตอบโต้การทำงานระหว่างวัตถุภายในระบบ โดยอยู่ในมุมมองลำดับของเวลา
- แผนภาพความร่วมมือ (collaboration diagram) ใช้สำหรับแสดงการตอบโต้การทำงานระหว่างวัตถุภายในระบบเช่นเดียวกับแผนภาพลำดับ แต่จะแสดงในมุมมองของโครงสร้างความสัมพันธ์ระหว่างวัตถุ
- แผนภาพสถานะ (statechart diagram) ใช้สำหรับแสดงสถานะการเปลี่ยนแปลงของระบบ โดยจะประกอบไปด้วยสถานะ เส้นการเปลี่ยนแปลง เหตุการณ์ และการกระทำ
- แผนภาพกิจกรรม (activity diagram) ใช้สำหรับแสดงฟังก์ชันการทำงานต่างๆ ของระบบในลักษณะของลำดับการไหลของการทำงานหนึ่งๆ ไปยังอีกการทำงานหนึ่ง
- แผนภาพส่วนประกอบ (component diagram) ใช้สำหรับอธิบายโครงสร้างของส่วนประกอบต่างๆ ว่าเชื่อมต่อกันอย่างไร และการทำงานของส่วนประกอบแต่ล่ะส่วนขึ้นอยู่กับส่วนประกอบส่วนอื่นๆ อย่างไร

- แผนภาพดิพลอยเมนท์ (deployment diagram) ใช้สำหรับอธิบายรูปแบบการจัดวางของระบบและส่วนประกอบต่างๆ ว่าต้องมีรูปแบบอย่างไรในการใช้งานจริง

ในงานวิจัยนี้เราได้ให้ความสนใจในแผนภาพสถานะ ซึ่งใช้สำหรับอธิบายพฤติกรรมของระบบ บอกสถานะทั้งหมดที่วัตถุสามารถเกิดขึ้นได้ และการเปลี่ยนแปลงสถานะของวัตถุเมื่อมีเหตุการณ์ต่างๆ เกิดขึ้น ซึ่งโดยทั่วไปแล้วแผนภาพสถานะแผนภาพหนึ่งจะใช้สำหรับอธิบายคลาสคลาสเดียว ปัจจุบันนี้มีการเขียนแผนภาพสถานะในหลายรูปแบบ แต่แผนภาพสถานะของยูเอ็มแอลนั้นมีพื้นฐานมาจากแผนภาพสถานะของเดวิด ฮาเรล [4] โดยรูปที่ 2.1 ข้างล่างนี้เป็นตัวอย่างระบบการสั่งซื้อสินค้าที่เขียนอธิบายด้วยแผนภาพสถานะ



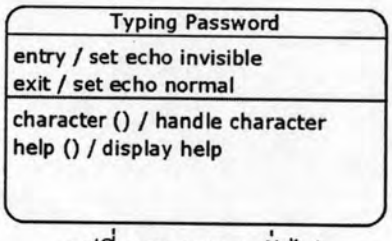
รูปที่ 2.1 แผนภาพสถานะแสดงระบบเครื่องโทรศัพท์

จากข้อกำหนดของแผนภาพนั้น ณ เวลาหนึ่งๆ ระบบจะต้องอยู่ในสถานะใดสถานะหนึ่งเท่านั้น หรือบางครั้งอยู่ในหลายสถานะในกรณีที่เป็นการทำงานแบบพร้อมกัน เมื่อมีเหตุการณ์ใดๆ เกิดขึ้นกับระบบแล้ว ระบบจะตอบรับต่อเหตุการณ์ที่เกิดขึ้นโดยการเปลี่ยนแปลงสถานะจากสถานะหนึ่งไปยังอีกสถานะหนึ่ง นอกจากนั้นเรายังอาจเพิ่มการอธิบายสิ่งที่ต้องกระทำ (actions) เข้าไปด้วยกับการเปลี่ยนแปลงที่เกิดขึ้น

แผนภาพสถานะยูเอ็มแอลประกอบด้วยสัญลักษณ์ที่ใช้สร้างแผนภาพดังต่อไปนี้

- สถานะทั่วไป (simple state) มีลักษณะเป็นสี่เหลี่ยมมุมโค้งดังรูปที่ 2.2 โดยภายในจะประกอบด้วยสองส่วน คือ
  - ชื่อสถานะ อยู่บริเวณตอนบนของสถานะ ซึ่งไม่สามารถมีสถานะที่มีชื่อเหมือนกันได้ภายในหนึ่งแผนภาพ

- การกระทำที่เกิดขึ้นภายในสถานะ อยู่บริเวณตอนล่างของสถานะ ส่วนนี้จะแสดงถึงสิ่งที่ต้องกระทำเมื่อระบบอยู่ในสถานะดังกล่าว ซึ่งสิ่งที่ต้องทำนี้สามารถระบุได้ทั้งหมดห้าชนิด คือ
  - *entry* ใช้สำหรับระบุสิ่งที่ต้องกระทำเมื่อระบบมีการเปลี่ยนแปลงเข้ามาสู่สถานะที่ระบุ
  - *exit* ใช้สำหรับระบุสิ่งที่ต้องกระทำเมื่อระบบมีการเปลี่ยนแปลงออกจากสถานะที่ระบุ
  - *do* ใช้สำหรับระบุถึงสิ่งที่ต้องกระทำอย่างต่อเนื่อง หรือจนเสร็จสมบูรณ์ในขณะที่ระบบอยู่ภายในสถานะที่ระบุ
  - *include* ใช้อ้างถึงแผนภาพย่อยของระบบ
  - ส่วนที่นอกเหนือจากสี่ชนิดข้างต้น คือ มีการระบุเหตุการณ์ เงื่อนไข และการกระทำ โดยเมื่อมีเหตุการณ์เกิดขึ้น และเงื่อนไขถูกต้อง จะกระทำการกระทำที่ระบุไว้ แต่สถานะจะไม่เปลี่ยนไปยังสถานะอื่น

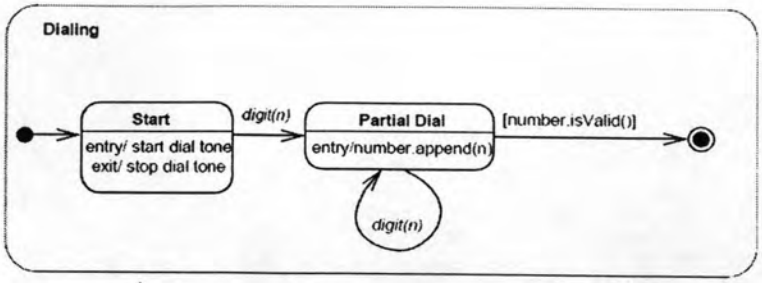


รูปที่ 2.2 สถานะทั่วไป

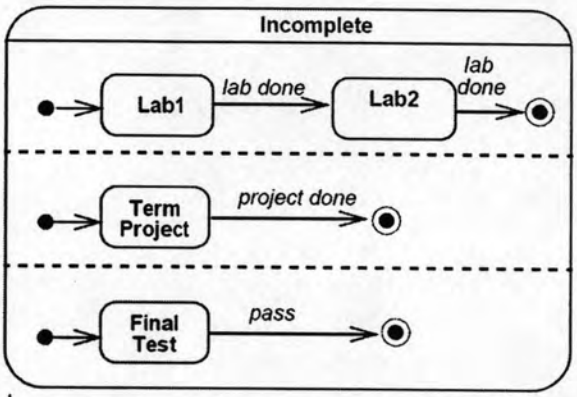
- สถานะสิ้นสุด (final state) มีลักษณะเป็นวงกลมทึบซึ่งมีวงกลมล้อมรอบอีกชั้น ดังรูปที่ 2.3 โดยใช้ระบุถึงการทำงานได้เสร็จเรียบร้อยแล้ว ซึ่งสถานะสิ้นสุดนี้ไม่สามารถมีเส้นการเปลี่ยนแปลงชี้ต่อไปยังสถานะอื่นได้

รูปที่ 2.3 สถานะเริ่มต้น

- สถานะประกอบ (composite state) มีลักษณะคล้ายกับสถานะทั่วไปแต่ภายในประกอบด้วยสถานะย่อยๆ เชื่อมต่อกัน หรือเป็นสถานะย่อยซึ่งทำงานพร้อมก็ได้ ดังรูปที่ 2.4 และรูปที่ 2.5

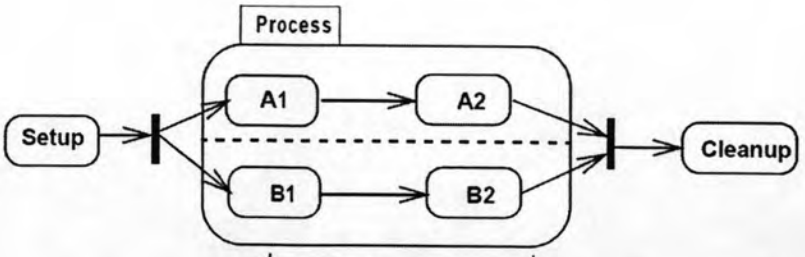


รูปที่ 2.4 สถานะประกอบแบบสถานะย่อยต่อเนื่อง



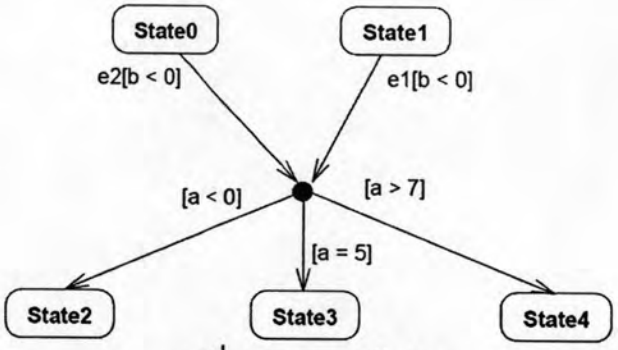
รูปที่ 2.5 สถานะประกอบแบบสถานย่อยทำงานพร้อมกัน

- เงื่อนไข (guard) เงื่อนไขจะมีลักษณะเป็นนิพจน์ที่ให้ความหมายเป็นถูกหรือผิดที่แสดงอยู่ในเส้นการเปลี่ยนแปลง หากให้ความหมายเป็นถูกเส้นการเปลี่ยนแปลงนั้นจะถูกอนุญาตให้เกิดการเปลี่ยนแปลงสถานะไปยังสถานะปลายทางได้ หากให้ความหมายเป็นผิดเส้นการเปลี่ยนแปลงจะไม่อนุญาตให้เกิดการเปลี่ยนสถานะ
- สถานะปลอม (pseudo state) เป็นสถานะแบบพิเศษเพื่อใช้ในการช่วยอธิบายพฤติกรรมของระบบ ซึ่งประกอบด้วย
  - สถานะเริ่มต้น (initial state) ใช้ระบุตำแหน่งเริ่มต้นของแผนภาพสถานะ มีลักษณะเป็นวงกลมทึบดังรูปที่ 2.5 และมีเส้นการเปลี่ยนแปลงหนึ่งเส้นเชื่อมโยงไปยังสถานะเริ่มต้นของระบบ
  - สถานะแยก (fork state) ใช้สำหรับแยกการทำงานไปสู่การทำงานแบบพร้อมกัน มีลักษณะเป็นเส้นตรงที่มีเส้นการเปลี่ยนแปลงขาเข้าหนึ่งเส้นและขาออกหลายเส้นดังรูปที่ 2.6



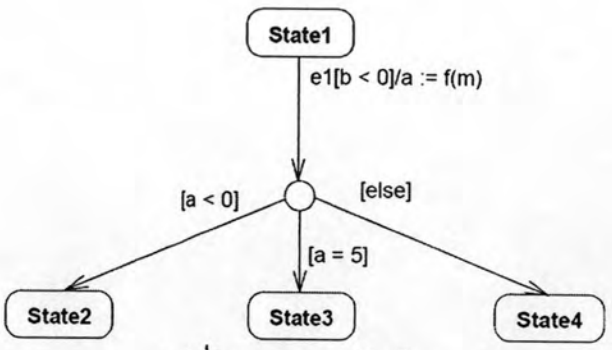
รูปที่ 2.6 สถานะแยกและเชื่อม

- สถานะเชื่อม (join state) ใช้สำหรับรวมเส้นการเปลี่ยนแปลงจากพื้นที่การทำงานพร้อมกันเข้าเป็นการทำงานเดียว มีลักษณะเป็นเส้นตรงที่มีเส้นการเปลี่ยนแปลงขาเข้าหลายเส้นและขาออกเพียงหนึ่งเส้นดังรูปที่ 2.6
- สถานะตัวต่อ (junction) ใช้สำหรับเป็นทางแยกหรือทางเชื่อมสำหรับเส้นการเปลี่ยนแปลงดังรูปที่ 2.7 โดยการเปลี่ยนแปลงสถานะจะพิจารณาเงื่อนไขในเส้นการเปลี่ยนแปลงทั้งหมดก่อนว่าถูกต้องหรือไม่ ถ้าถูกต้องการเปลี่ยนสถานะจะเปลี่ยนจากสถานะต้นทางไปยังสถานะปลายทางโดยไม่หยุดที่สถานะตัวต่อก่อน



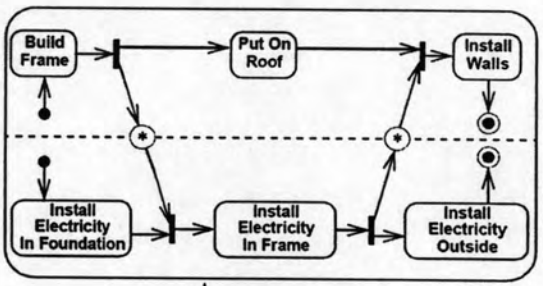
รูปที่ 2.7 สถานะตัวต่อ

- สถานะตัวเลือก (choice) ใช้สำหรับเป็นทางแยกหรือทางเชื่อมสำหรับเส้นการเปลี่ยนแปลงดังรูปที่ 2.8 โดยการพิจารณาเงื่อนไขในเส้นการเปลี่ยนแปลงหลังสถานะตัวเลือกจะถูกพิจารณาหลังจากสถานะของระบบได้เปลี่ยนมาอยู่ที่สถานะตัวเลือกเรียบร้อยแล้ว ในกรณีที่ตัวเลือกเส้นการเปลี่ยนแปลงหลังจากสถานะตัวเลือกนั้นไม่มีตัวใดมีเงื่อนไขถูกต้องและทำให้ไม่สามารถเปลี่ยนสถานะจากสถานะตัวเลือกไปยังสถานะปลายทางได้ จะถือว่าแผนภาพนั้นมีความบกพร่อง



รูปที่ 2.8 สถานะตัวเลือก

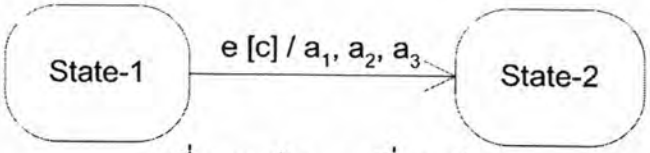
- สถานะซิง (synch state) เป็นสถานะแบบพิเศษที่หมายถึงการทำงานพร้อมกันระหว่างสองระบบที่ทำงานพร้อมกัน โดยจะมีลักษณะเป็นวงกลมวางทับอยู่บนเส้นแบ่งระหว่างระบบที่ทำงานพร้อมกันและมีจำนวนระบุภายในเพื่อแสดงความจุดังรูปที่ 2.9 จำนวนที่ระบุภายในนั้นอาจเป็นตัวเลขจำนวนบวก หรือดอกจันทน์ก็ได้ โดยดอกจันทน์จะหมายถึงความจุที่ไม่จำกัดจำนวน



รูปที่ 2.9 สถานะซิง

- เส้นการเปลี่ยนแปลง (transition) เส้นการเปลี่ยนแปลงจะมีลักษณะเป็นลูกศรเชื่อมต่อระหว่างสองสถานะ และมีคำอธิบายบนเส้นซึ่งประกอบด้วย เหตุการณ์, เงื่อนไข และสิ่งที่ต้องกระทำ ดังรูปที่ 2.10 ซึ่งอธิบายถึงว่า ถ้าเหตุการณ์ที่ระบุไว้เกิดขึ้นและเงื่อนไขเป็นจริง สถานะของระบบจะถูกเปลี่ยนจากสถานะต้นทางไปยังสถานะปลายทาง และประมวลผลการกระทำที่ระบุไว้ ลักษณะของคำอธิบายเส้นการเปลี่ยนแปลงเป็นดังนิพจน์ข้างล่าง

event – name '(' *comma – separated – parameter – list* ')' '[' *guard – condition* ']' '/' *action – ex pression*



รูปที่ 2.10 เส้นการเปลี่ยนแปลง

2.1.2 นิพจน์สม่ำเสมอ (Regular Expression)

นิพจน์สม่ำเสมอประกอบด้วยตัวอักษร (alphabet) อักษรว่าง (ε) และตัวดำเนินการทั้งหมด 3 ตัว คือ เชื่อมต่อ ($\cdot$) ทางเลือก ($+$) และคลีนสตาร์ (*) ซึ่งตัวดำเนินการแต่ละตัวเมื่อถูกนำมาใช้ในการเชื่อมต่อตัวอักษรมีความหมายดังนี้

- ตัวดำเนินการเชื่อมต่อ เมื่อเชื่อมระหว่างอักษรใดๆ ลำดับจะมีความสำคัญ คือ เหตุการณ์ที่อักษรที่อยู่ด้านซ้ายจะเกิดก่อนอักษรที่อยู่ด้านขวาของตัวดำเนินการ เช่น $a \cdot b = \{a \cdot b\}$ เราสามารถใช้ตัวเลขยกกำลังในกรณีที่ต้องการเชื่อมตัวอักษรตัวเดียวกันซ้ำกันหลายๆ ครั้งได้ เช่น $a^1 = \{a\}$, $a^2 = \{a \cdot a\}$ ในบางกรณีสามารถละเว้นการเขียนตัวดำเนินการ $\cdot$ ได้หากไม่ทำให้เกิดความสับสน เช่น $a \cdot b = ab$
- ตัวดำเนินการทางเลือก เมื่อเชื่อมระหว่างอักษรใดๆ หมายถึงเหตุการณ์ที่สามารถเกิดตัวอักษรใดก็ได้ เช่น $a + b = \{a, b\}$ ซึ่งตัวดำเนินการทางเลือกนี้มีคุณสมบัติของการสลับข้างได้ คือ $a + b = b + a$
- ตัวดำเนินการคลีนสตาร์ เมื่อใช้กับนิพจน์ใดๆ จะหมายถึงการรวมกันของการทำซ้ำของนิพจน์ตั้งแต่ 0 ครั้ง ไปเรื่อยๆ ไม่สิ้นสุด คือ $A^* = \bigcup_{i=0,1,\ldots} A^i$ ตัวอย่างเช่น

$$a^* = a^0 + a^1 + a^2 + \ldots = \{\varepsilon, a, aa, \ldots\}$$

2.1.3 ซีอาร์อี (CRE: Concurrent Regular Expression)

ซีอาร์อีเป็นส่วนเพิ่มเติมการทำงานของนิพจน์สม่ำเสมอ เพื่อช่วยให้นิพจน์สม่ำเสมอสามารถอธิบายระบบที่งานแบบพร้อมกันได้ โดยซีอาร์อีได้เพิ่มตัวดำเนินการเข้ามาอีก 4 ตัว คือ

- Interleaving ใช้สัญลักษณ์แทนคือ $||$ โดยใช้สำหรับอธิบายการแทรกสลับกันของสองระบบที่ทำงานพร้อมกันอย่างอิสระ เช่น $a || b = \{ab, ba\}$ หรือ $ab || cd = \{abcd, acbd, cabd, cdab\}$ โดยถ้าหากเป็นการทำ interleaving ซ้ำ

ตัวเองจะใช้เครื่องหมายคล้ายกับยกกำลังแต่มีวงเล็บล้อมรอบเพื่อให้แตกต่างกัน คือ $A^{(2)} = A \parallel A$

- Alpha-closure ใช้สัญลักษณ์แทน คือ $^{\alpha}$ โดยความหมายคือใช้อธิบายการรวมกันของการทำ interleaving กับตัวเองตั้งแต่ 0 ครั้งไปเรื่อยๆ ไม่มีที่สิ้นสุด คือ $A^{\alpha} = \bigcup_{i=0,1,...} A^{(i)}$ ตัวอย่างเช่น

$$(ab)^{\alpha} = (ab)^{(0)} + (ab)^{(1)} + (ab)^{2} + ... = \bigcup_{i=0,1,...} (ab)^{(i)}$$

- Synchronous Composition ใช้อธิบายการซิงโครนัสกันของระบบ 2 ระบบที่ทำงานพร้อมกัน เช่น $ab[\,]bc = \{abc\}$, $ab[\,]ac = \{abc, acb\}$ หรือ $ab[\,]ba = \{\,\}$ ในกรณีที่ระบบ 2 ระบบไม่มีส่วนที่ซิงโครนัสกันเราสามารถแทนที่ตัวดำเนินการด้วยตัวดำเนินการ interleaving ได้ เช่น $ab[\,]cd = ab \parallel cd$
- Renaming ใช้ช่วยสำหรับการเปลี่ยนชื่อตัวอักษรในนิพจน์

2.1.4 ไพแคลคูลัส (π-Calculus)

ไพแคลคูลัสเป็นแคลคูลัสที่อยู่บนพื้นฐานของการสื่อสารระหว่างกระบวนการ (process) การสื่อสารจะกระทำผ่านช่องที่ได้กำหนดไว้ โดยที่กระบวนการที่เป็นฝ่ายส่งค่า v ผ่านช่อง c จะใช้สัญลักษณ์แทน คือ $\bar{c}\langle v\rangle$ ซึ่งอาจกระทำการสื่อสารกับอีกกระบวนการหนึ่งซึ่งเป็นฝ่ายรับผ่านช่อง c โดยเก็บค่าไว้ใน w คือ $c(w)\cdot P$ และเปลี่ยนคุณสมบัติตัวเองเป็น P โดยความแตกต่างของไพแคลคูลัสจากแคลคูลัสของกระบวนอื่นๆ คือ สามารถสร้างช่องทางการสื่อสารใหม่ให้กับตัวกระบวนการได้โดยการรับค่าช่องทางการสื่อสารผ่านการสื่อสารกับกระบวนการอื่นๆ ซึ่งคุณสมบัติดังกล่าวนี้ส่งผลให้ความสามารถในการอธิบายคุณสมบัติต่างๆ ของระบบที่มีอยู่สูงขึ้น

วากยสัมพันธ์ของไพแคลคูลัสประกอบด้วยส่วนต่างๆ ดังนี้

- 0 หมายถึง ว่าง (Null) คือ กระบวนการที่ไม่มีการทำงาน
- $x(\vec{y})\cdot P$ หมายถึง การรับค่า $y_1, y_2, ..., y_n$ จากช่อง x และเปลี่ยนคุณสมบัติตัวเองเป็น P กรณีของ $x(\vec{y})\cdot 0$ สามารถเขียนในรูป $x(\vec{y})$ ได้โดยละเว้น 0 ไว้ในฐานที่เข้าใจ
- $\bar{x}\langle\vec{y}\rangle\cdot P$ หมายถึง การส่งค่า $y_1, y_2, ..., y_n$ ไปทางช่อง x และเปลี่ยนคุณสมบัติตัวเองเป็น P กรณีของ $\bar{x}\langle\vec{y}\rangle\cdot 0$ ก็เช่นกันสามารถเขียนในรูป $\bar{x}\langle\vec{y}\rangle$ ได้โดยละเว้น 0 ไว้ในฐานที่เข้าใจ
- $P \mid Q$ หมายถึง การทำงานพร้อมกันของกระบวนการ P และ Q

- $P+Q$ หมายถึง ทางเลือกเชิงไม่กำหนด (non-deterministic choice) ระหว่าง P หรือ Q
- $new\vec{x}$ หมายถึง การสร้างช่องสื่อสาร $x_1, x_2, ..., x_n$ ขึ้นมาใหม่ โดยช่องสื่อสารดังกล่าวจะถูกใช้งานเฉพาะในกลุ่มที่กำหนดเท่านั้น ระบบภายนอกจะไม่สามารถมองเห็นช่องสื่อสารดังกล่าว
- $!P$ หมายถึง การทำงานพร้อมกันจำนวนอนันต์ของ P คือ $!P = P|!P$
- $A(x_1, x_2, ..., x_n) = ...$ หมายถึง A ประกอบด้วย n พารามิเตอร์
- $[x=y]P$ หมายถึง ระบบจะมีคุณสมบัติเป็น P ก็ต่อเมื่อ $x=y$ ถ้าไม่จะเป็นว่าง
- τ หมายถึง การกระทำที่เกิดขึ้นภายใน ซึ่งเกิดจากการรับและส่งค่ากันเองภายในระบบ
- $P\{\vec{y}/\vec{x}\}$ หมายถึง การแทนที่ $\vec{x}$ ใน P ด้วย $\vec{y}$

ไพแคลคูลัสมีคุณสมบัติที่เรียกว่า ความสัมพันธ์ของการลดทอน (reduction relation) คือ ความสัมพันธ์ระหว่างกระบวนการ $P \longrightarrow Q$ โดยที่กระบวนการ P สามารถเปลี่ยนไปเป็นกระบวนการ Q ด้วยการประมวลผลเพียง 1 ขั้นตอน หรือการสื่อสารกันระหว่างตัวรับกับตัวส่ง ยกตัวอย่าง เช่น

- $a \cdot P \xrightarrow{a} P$ เป็นการประมวลผล a จากนั้นกระบวนการเปลี่ยนจาก $a \cdot P$ ไปเป็น P
- $\overline{a} \cdot P \xrightarrow{\overline{a}} P$ เป็นการประมวลผล $\overline{a}$ จากนั้นกระบวนการเปลี่ยนจาก $\overline{a} \cdot P$ ไปเป็น P
- $a \cdot P | \overline{a} \cdot Q \xrightarrow{a} P | \overline{a} \cdot Q$ เป็นการประมวลผล a จากนั้นกระบวนการเปลี่ยนจาก $a \cdot P | \overline{a} \cdot Q$ ไปเป็น $P | \overline{a} \cdot Q$
- $a \cdot P | \overline{a} \cdot Q \xrightarrow{\overline{a}} a \cdot P | Q$ เป็นการประมวลผล $\overline{a}$ จากนั้นกระบวนการเปลี่ยนจาก $a \cdot P | \overline{a} \cdot Q$ ไปเป็น $a \cdot P | Q$
- $a \cdot P | \overline{a} \cdot Q \xrightarrow{\tau} P | Q$ เป็นการประมวลผล τ คือ a และ $\overline{a}$ ถูกประมวลผลพร้อมกัน โดย $\overline{a}$ เป็นฝ่ายส่ง และ a เป็นฝ่ายรับ จากนั้นกระบวนการเปลี่ยนจาก $a \cdot P | \overline{a} \cdot Q$ ไปเป็น $P | Q$
- $new\, a(a \cdot P | \overline{a} \cdot Q) \xrightarrow{\tau} new\, a(P | Q)$ เป็นการประมวลผล τ คือ a และ $\overline{a}$ ถูกประมวลผลพร้อมกัน โดย $\overline{a}$ เป็นฝ่ายส่ง และ a เป็นฝ่ายรับ จากนั้นกระบวนการเปลี่ยนจาก $new\, a(a \cdot P | \overline{a} \cdot Q)$ ไปเป็น $new\, a(P | Q)$

จากตัวอย่างของการลดทอนจะเห็นได้ว่า $a \cdot P \mid \overline{a} \cdot Q$ สามารถเกิดการประมวลผลลดทอนได้ทั้งหมด 3 รูปแบบ คือ ประมวลผล a จะได้ $a \cdot P \mid \overline{a} \cdot Q \xrightarrow{a} P \mid \overline{a} \cdot Q$ ประมวลผล $\overline{a}$ จะได้ $a \cdot P \mid \overline{a} \cdot Q \xrightarrow{\overline{a}} a \cdot P \mid Q$ หรือประมวลผล τ จะได้ $a \cdot P \mid \overline{a} \cdot Q \xrightarrow{\tau} P \mid Q$ โดยการประมวลผลลดทอน a และ $\overline{a}$ นั้นเปรียบเสมือนระบบเกิดการสื่อสารคู่กับ $\overline{a}$ และ a ของสิ่งแวดล้อมภายนอกตามลำดับ แต่ τ นั้นเกิดจากการสื่อสารภายในของ a ใน $a \cdot P$ คู่กับ $\overline{a}$ ใน $\overline{a} \cdot Q$ หากเราต้องการให้ช่องการสื่อสาร a เกิดการสื่อสารกันเฉพาะภายในระบบเท่านั้นและไม่สามารถสื่อสารกับสิ่งแวดล้อมได้ เราสามารถซ่อนช่องสื่อสาร a จากระบบภายนอกได้โดยการเติม $newa$ คลุมระบบที่ต้องการ ดังตัวอย่าง $newa(a.P \mid \overline{a}.Q)$ จะหมายถึงช่องสัญญาณ a เป็นช่องสื่อสารที่รู้จักเฉพาะภายในระบบเท่านั้น ระบบภายนอกไม่สามารถสื่อสารกับ $a \cdot P \mid \overline{a} \cdot Q$ ผ่านช่องทาง a ได้

2.2 เอกสารและงานวิจัยที่เกี่ยวข้อง

2.2.1 "Formalizing sequence diagrams and state machines using Concurrent Regular Expression" โดย มิตซูทากะ โอกาซากิ, โทชิอากิ อาโอกิ และทาคูระ คาทายามะ [9]

งานวิจัยชิ้นนี้ได้เสนอวิธีการแปลงแผนภาพลำดับและแผนภาพสถานะไปเป็นซีอาร์อี และได้เสนอวิธีการสร้างแผนภาพสถานะจากแผนภาพลำดับ โดยการใช้ภาษาซีอาร์อีเป็นตัวกลางในการแปลง และได้เสนอวิธีการตรวจสอบความต้องกันระหว่างแผนภาพทั้งสองชนิดด้วยองค์ประกอบที่ถูกนำมาพิจารณาในส่วนของแผนภาพสถานะนั้นมีเพียงสถานะทั่วไปและเส้นการเปลี่ยนเท่านั้น ยังไม่ครอบคลุมองค์ประกอบของแผนภาพสถานะยูเอ็มแอลอีกจำนวนมาก

2.2.2 "Towards Formalizing UML State Diagrams in CSP" โดย มวน ยองและ ไมเคิล บัทเลอร์ [8]

งานวิจัยนี้ได้เสนอวิธีการแปลงแผนภาพสถานะไปเป็นซีเอสพี (CSP: Communicating and Sequential Processes) [10] ซึ่งเป็นแคลคูลัสของกระบวนการเช่นกัน โดยวิธีการแปลงนั้นได้เปรียบเทียบ สถานะของแผนภาพไปเป็นกระบวนการซีเอสพี และเหตุการณ์ของแผนภาพไปเป็นเหตุการณ์ของซีเอสพี และได้สร้างกฎการแปลงขึ้นมาสำหรับองค์ประกอบแต่ละตัวในแผนภาพและการเชื่อมต่อในแบบต่างๆ รวมถึงได้สร้างเครื่องมือในการแปลงแผนภาพสถานะไปเป็นภาษาซีเอสพีอย่างอัตโนมัติ ซึ่งภาษาซีเอสพีนี้สามารถนำเข้าไปยังเอฟดีอาร์ (FDR: Failures Divergence Refinement) [12] ซึ่งเป็นเครื่องมือตรวจสอบการแบ่งละเอียด (refinement checker) ได้องค์ประกอบในการสร้างแผนภาพสถานะที่งานวิจัยนี้ได้เสนอกฎการแปลงนั้นยังขาดในส่วนของ

สถานะประกอบที่สถานะย่อยภายในเป็นแบบทำงานพร้อมกัน ซึ่งมีความสำคัญมากในการใช้อธิบายระบบที่มีการทำงานแบบพร้อมกัน

2.2.3 "Extraction of π-Calculus specifications from UML sequence and state diagrams" โดย แคทรีนา คอเรนบลาท และคอแรโด พริอามี [5]

งานวิจัยนี้ได้เสนอวิธีการแปลงข้อกำหนดจากแผนภาพยูเอ็มแอลสถานะและลำดับไปเป็นไพแคลคูลัส โดยได้อาศัยข้อมูลจากแผนภาพลำดับเป็นตัวหลักและใช้ข้อมูลจากแผนภาพสถานะช่วยเสริมในส่วนรายละเอียด โดยองค์ประกอบของแผนภาพลำดับนั้นได้ถูกกำหนดวิธีการแปลงได้ครอบคลุมส่วนหลักๆ ทั้งหมด แต่ในส่วนของแผนภาพสถานะนั้นมีเพียงส่วนหลักสองส่วนเท่านั้น คือ สถานะ และเส้นการเปลี่ยนแปลง

2.2.4 "Symbolic Model Checking of UML Statechart Diagrams with an Integrated Approach" โดย วิตัส เอส ดับบิว แลม และจูเลียน แพดเจท [6]

งานวิจัยนี้ได้เสนอวิธีการแปลงแผนภาพสถานะยูเอ็มแอลไปเป็นไพแคลคูลัสอย่างเป็นระบบ ซึ่งส่วนการแปลงที่ได้รับนำเสนอนั้นจะครอบคลุมส่วนหลักๆ ขององค์ประกอบในการสร้างแผนภาพ และได้ทำการแปลงจากไพแคลคูคัสเป็นข้อมูลนำเข้าเพื่อใช้กับ NuSMV [2] ซึ่งเป็นเครื่องมือในการตรวจสอบต้นแบบ (model checker) ชนิดหนึ่ง แต่ทั้งนี้ผลที่ได้จากการแปลงแผนภาพไปเป็นไพแคลคูลัสไม่ได้อธิบายถึงความสัมพันธ์ระหว่างแผนภาพอื่นๆ ภายในระบบเดียวกัน ทำให้การตรวจสอบระบบสามารถทำได้เฉพาะภายในแผนภาพของแต่ละวัตถุเท่านั้น ไม่สามารถตรวจสอบภาพรวมการทำงานของระบบซึ่งมีวัตถุหลายๆ วัตถุทำงานด้วยกัน